REPORT DOCUMENTATION PAGE

AFRL-SR-AR-TR-07-0411

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
15/01/2007	Final Report	01/12/2005---30/11/2006

4. TITLE AND SUBTITLE

Algorithm and Implementation of *P*-adic Cyclic Codes Using Exact Arithmetic Library Developed for Quantum Computing

5a. CONTRACT NUMBER

5b. GRANT NUMBER

FA9550-06-1-0038

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Chao Lu

Computer & Information Sciences

Towson University

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Towson University
8000 York Road
Towson, MD 21252

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Dr. Jon Sjogren
AFOSR/NM
4015 Wilson Blvd, Room 713
Arlington, VA 22203-1954

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Distribution A: Approved for Public Release

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Summary: The first part of the research is that we have expanded the Exact Scientific Computational Library (ESCL), and Dixon's algorithm on rational *N by N* matrix inverse was implemented. We studied and experimented the relation of required length *M* of *p*-adic expansion and the prime *p*, and the possible use of the length of periodicity of a rational number's *p*-adic expansion in determining the length of required *M* in rational matrix operations.

The second part of the work is to develop and implement computational algorithms for *p*-adic cyclic code generation, which is based on the results of the paper, *Modular and p-adic cyclic codes, by A.R.Calderbank and N.J.A. Sloane*. Algorithms and software have been developed to give an alternative solution to factorize the polynomial X^n-1 over the ring of integers modulo p^a, where *p* is a prime not dividing *n*, and it can generate the table of cyclic codes using the divisors of X^n-1 as their generator polynomials.

All the implementation of ESCL is in C++, as well as the software to generate *p*-adic cyclic codes.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Chao Lu
					19b. TELEPHONE NUMBER *(include area code)* 410-704-3701

Algorithms and Implementation for *P*-adic Cyclic Codes Using Exact Arithmetic Library Developed for Quantum Computing

Final Report (Grant # **FA9550-06-1-0038***)*

Chao Lu

Computer & Information Sciences

Towson University

January 16, 2007

1. Summary

The first part of the research is that we have expanded the Exact Scientific Computational Library (ESCL), and the Dixon's algorithm [1] on rational *N by N* matrix inverse was implemented. We studied the relation of required length *M* of *p*-adic expansion and the prime *p* by experiments, and the possible use of the length of periodicity of a rational number's *p*-adic expansion in determining the length of required *M* in rational matrix operations.

The second part of the work is to develop and implement computational algorithms for *p*-adic cyclic code generation, which is based on the results of the paper, *Modular and p-adic cyclic codes, by A.R.Calderbank and N.J.A. Sloane* [2]. Algorithms and software have been developed to give an alternative solution to factorize the polynomial X^n-1 over the ring of integers modulo p^a, where *p* is a prime not dividing *n*, and it can generate the table of cyclic codes using the divisors of X^n-1 as their generator polynomials.

All the implementation of ESCL is in C++, as well as the software to generate *p*-adic cyclic codes.

2. Progress on ESCL Using *P*-adic Arithmetic

In the design of ESCL we employ several developing tools and a specific library--NTL. We use Visual C++ 6.0 and MFC to develop and debug the codes. MATLAB Symbolic is used for comparison. We adopt the NTL library to process the input and output of arbitrary long integers, which are the numerator and denominator of a rational NTL, written by Shoup [3], a high-performance, portable C++ library for number theory that provides both data structures and algorithms for arbitrary length integers. NTL allows manipulation of integers for vectors, matrices, and polynomials over finite fields, and arbitrary precision floating-point arithmetic.

Our ESCL has the following operations:

1) Converting rational to *p*-adic number and via verse.
2) Single rational number's calculation: addition, subtraction, multiplication and division.
3) The addition, subtraction and multiplication of two rational matrixes.
4) The inverse of a single rational *N by N* matrix.

In order to improve the efficiency of ESCL, a proper prime *p* and the length *M* of *p*-adic expansion need to be determined. We did experiments on a random rational matrix with different prime numbers and record the execution time. Our experiments show that the running time is more related to the length *M* of *p*-adic sequence, where "*M*" can be calculated based on the prime *p*:

$$m = 2\left\lfloor \log(\delta\lambda^{-1}) / \log p \right\rfloor, \qquad (1)$$

where the $\delta = \prod \lambda_i$ and λ_i is the *Euclidean* length of the *ith* column.

The process of choosing *p* and *M* can be done as follows. Suppose that for certain "*p*", we can get "*M*" very

small. If we choose a small M, then use (1) to calculate prime p. Thus we will get a relatively big prime and a theoretic smallest M, and some adjustment is needed during the process. The flow chart is shown in Figure 1.

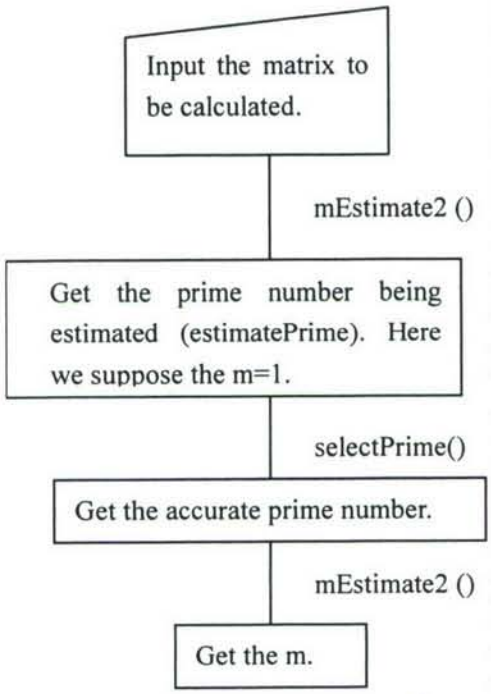

Figure 1. Estimating the prime number

Dixon's algorithm for matrix inverse from "Exact Solution of Linear Equations Using *P*-adic Expansions" [1] was implemented, which is the newly added function since the last report. To get the A^{-1}, for which $AxA^{-1}=I$, we take the following three steps.

1) Get the nxn matrix C, whose entries lie in *[0, p-1]*

   $AA^{-1}=I \Rightarrow AA^{-1} \bmod p \equiv I \bmod p \Rightarrow A\,C \equiv I \bmod p; \Rightarrow C \equiv A^{-1} \bmod p.$

2) Compute a p-adic approximation x' to A^{-1}. x' is a nx1 column of A^{-1}.

$$x' = \sum_{i=0}^{m-1} x_i p^i = \sum_{i=0}^{m-1} (Cb_i \bmod p) p^i$$

3) Transfer x' to rational field.

Repeat doing 2) 3), until *bi=bn-1*,

and $b_0=\begin{bmatrix}1\\0\\..\\0\end{bmatrix}$ $b_1=\begin{bmatrix}0\\1\\..\\0\end{bmatrix}$ $\ldots\ b_{n-1}=\begin{bmatrix}0\\0\\..\\1\end{bmatrix}$

Figure 2 shows an example for this operation.

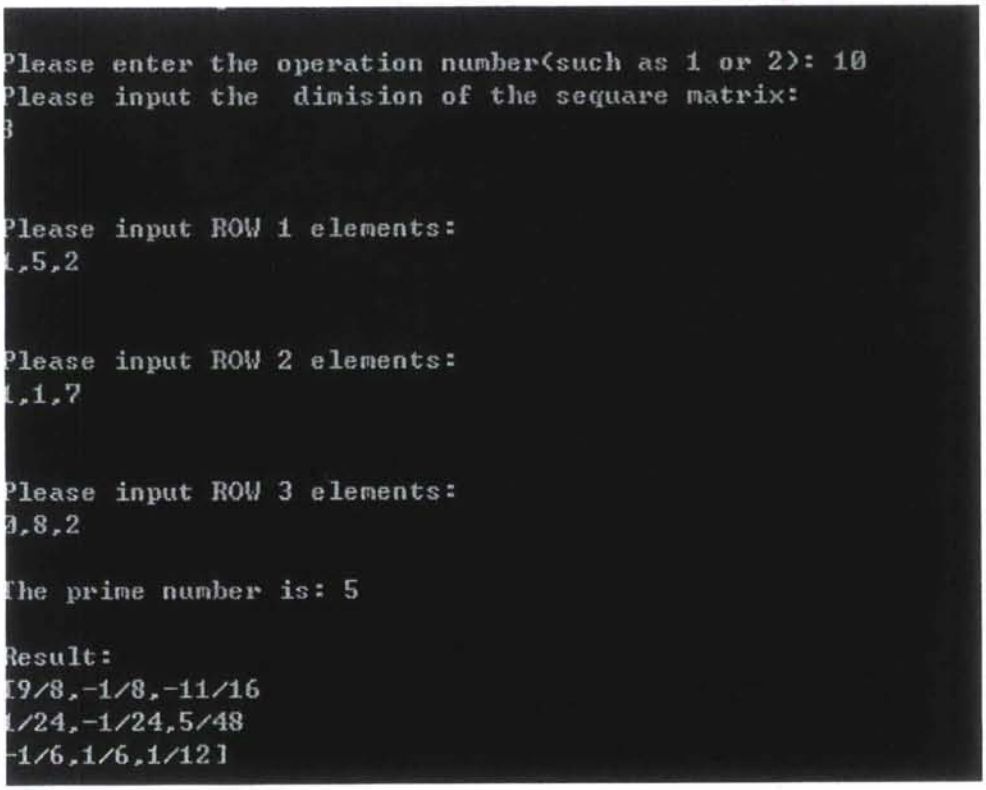

Figure 2. Matrix inverse example

Our system is capable of carrying out exact arithmetic operations for rational numbers using the finite segmented p-adic numbers. The representation of the rational numbers has several appealing properties. Given the usual representation of the positive integers, it is, in a genuine sense, the natural extension to the rational. The algorithms for addition, subtraction, and multiplication are those of the usual integer arithmetic; the division algorithm is truly the analogue of multiplication.

When two *p-adic* rational numbers are added, subtracted, multiplied, or divided, the result is an infinite but eventually repeating sequence of digits (periodicity).

Despite the advantages of finite segment p-adic arithmetic and exact representation, there are certain issues need to be addressed:

1) Improve the EstimateM(). Currently, the M estimation does not always work well. In some situation, it cannot give the sufficient number of digits for p-adic sequence for exact computation. To avoid this kind of errors, we have increased its size to 30 and more.
2) A more efficient method for prime number selection for the given rational.

3. The Length of Periodicity for the *P*-adic Expansion

To determine what is the sufficient length M of p-adic expansion for a rational number in the matrix operation is not a trivial problem. Let us observe what happens after the arithmetic operations of two p-adic

sequences.

All rational numbers can be uniquely written in the form of $a = \sum_{j=0}^{\infty} a_j p^j$.

We know that a real number is rational if and only if its decimal expansion is periodic. Similarly, a *p*-adic number is rational if and only if its *p*-adic expansion is periodic. Consequently, since we are primarily interested in the *p*-adic expansions of rational numbers we will be dealing only with *p*-adic expansions which are periodic.

The expansion eventually repeats to the right. That is, if a is a rational number, then it has a repeating pattern of $a_j p^j$ in its *P*-adic expansion, i.e., it is of the form

$$a = .A_0 \ldots A_s \overline{a_0 \ldots a_{n-1}}$$

For example:

$\frac{1}{3} = .2\overline{31}$ (p=5)

$\frac{2}{3} = .4\overline{13}$ (p=5)

The operation of addition, subtraction, multiplication and division in the set of *p*-adic numbers are quite similar to the corresponding operations in decimals. The main difference, however, is that we proceed from "left to right" rather than from "right to left" as we do with decimals.

Addition/subtraction

Assume that we have two P-adic sequences: (s<t)

$$a = .A_1 \ldots A_s \overline{a_1 \ldots a_n}$$

$$b = .B_1 \ldots B_t \overline{b_1 \ldots b_m}$$

Line up these two sequences

$$\begin{array}{l|l} A_1 A_2 \ldots A_s a_1 \ldots a_{t-s} & a_{t-s+1} \cdots \\ B_1 \qquad\qquad \ldots B_t & b_1 \ldots \end{array}$$

Set $a_{t-s+1} = c_1$

$\because a_{t-s+1+n} = a_{t-s+1}$

$\therefore c_{1+n} = c_1$

$$\begin{array}{l|l} A_1A_2...A_sa_1...a_{t-s} & c_1...c_nc_1...c_n... \\ B_1 \quad\quad ...B_t & b_1...b_mb_1...b_m... \end{array}$$

Let's consider the right side of the vertical stroke line

	c_1	c_2	c_3	...	c_n	c_1	...	...	c_n...
+	b_1	b_2	b_3	...	...	...	b_m	b_1	...	b_m...
	c_1+b_1	c_2+b_2	c_3+b_3	...	c_k+b_k...				

Suppose two integer x,y satisfy the condition that $x \neq y$ and $c_x+b_x = c_y+b_y$

$\because$ c_x and b_y can be any number

$\therefore$ in the worst case, we must make sure that $c_x = a_y = a_i$ and $c_x = b_y = b_j$

$x = k1 \times n + i = k2 \times m + j$

$y = k3 \times n + i = k4 \times m + j$

$\Rightarrow$ $x\text{-}y = (k1\text{-}k3) \times n = (k2\text{-}k4) \times m$

$\because x \neq y$

$\therefore$ *k1-k3 and k2-k4* are nonzero integers

x-y must be exactly divisible by *n* and *m*, the smallest integer which satisfies this requirement is *LCM(n,m)*.

The carry digits' modification

We only need to consider the *(t+1)th* carry digit, because it will cause the exception of the periodic part. Since the maximum carry digit of the addition of two numbers is 1, we can classify the following numbers into three types and they have different effects.

Type One: The *(t+1)th* digit is less than *p-1*, assume the *(t+1)th* digit is *q*:

$$\because q < p-1 \Rightarrow q+1 < p$$

$$\Rightarrow .q...+1 \text{ (the carry digit)} = .(q+1)...$$

so only one digit will be affected.

Type Two: The next r digits following the t*th* digit have the value *p-1*

Assume the (t+r+1)*th* digit is q, and $q < p-1$

$$.\overbrace{(p-1)...(p-1)}^{r}q...+1... = .\overbrace{0...0}^{r}(q+1)...$$

$r+1$ digits will be affected.

In addition, $r < LCM(n,m)$

Because if $r = LCM(n,m)$, then it belongs to the type three.

Type Three: The digits following the t*th* digit are rotation numbers of *p-1*

$$.\overline{p-1}+1=.\bar{0}=0$$

The result is 0.

We can draw a conclusion that the maximum length of the p-adic expansion is:

$$2\times LCM(n,m)+\max(s,t)-1.$$

Multiplication

$$a\times b=.A_1\dots A_s\overline{a_1\dots a_n}\times .B_1\dots B_t\overline{b_1\dots b_m}$$

$$=.A_1\dots A_s\overline{a_1\dots a_n}\times .B_1\dots B_t+.A_1\dots A_s\overline{a_1\dots a_n}\times .\overbrace{0\dots 0}^{t}\overline{b_1\dots b_m}$$

$$=\underbrace{.A_1\dots A_s\overline{a_1\dots a_n}\times .B_1\dots B_t}_{1}+\underbrace{.A_1\dots A_s\times .\overbrace{0\dots 0}^{t}\overline{b_1\dots b_m}}_{2}+\underbrace{.\overbrace{0\dots 0}^{s}\overline{a_1\dots a_n}\times .\overbrace{0\dots 0}^{t}\overline{b_1\dots b_m}}_{3}$$

Let's analyze the three parts separately,

Part 1:

$$\because .A_1\dots A_s\in Z$$

$$\therefore .A_1\dots A_s\overline{a_1\dots a_n}\times .B_1\dots B_t=.C_1\dots C_u\overline{c_1\dots c_n}$$

That is, the result is a p-adic sequence, which has a periodic part of n digits.

Part 2:

$$\because .B_1\dots B_t\in Z$$

$$\therefore .A_1\dots A_s\times .\overbrace{0\dots 0}^{t}\overline{b_1\dots b_m}=.D_1\dots D_v\overline{d_1\dots d_m}$$

That is, the result is a p-adic sequence, which has a periodic part of m digits.

Part 1 + Part 2:

$$\underbrace{.A_1\dots A_s\overline{a_1\dots a_n}\times .B_1\dots B_t}_{1}+\underbrace{.A_1\dots A_s\times .\overbrace{0\dots 0}^{t}\overline{b_1\dots b_m}}_{2}$$

According to the previous conclusion we get from the Addition part,

$$\Rightarrow .A_1\dots A_s\overline{a_1\dots a_n}\times .B_1\dots B_t+.A_1\dots A_s\times .\overbrace{0\dots 0}^{t}\overline{b_1\dots b_m}=.E_1\dots E_w\overline{e_1\dots e_{LCM(n,m)}}$$

This is a p-adic sequence which has a periodic part of $LCM(n,m)$ digits.

Part 3:

$$\overbrace{.0\ldots0a_1\ldots a_n}^{s} \times \overbrace{.0\ldots0b_1\ldots b_m}^{t}$$

$$= (.a_1\ldots a_n)\times(1+p^n+p^{2n}+\ldots+p^{\infty})\times(.b_1\ldots b_m)\times(1+p^m+p^{2m}+\ldots+p^{\infty})\times p^{s+t}$$

$$= \underbrace{(.a_1\ldots a_n)\times(.b_1\ldots b_m)\times p^{s+t}}_{4} \times \underbrace{(1+p^n+p^{2n}+\ldots+p^{\infty})\times(1+p^m+p^{2m}+\ldots+p^{\infty})}_{5}$$

Part 4 $(.a_1\ldots a_n)\times(.b_1\ldots b_m)\times p^{s+t} \in Z$.

Part 5 is the one we need to focus on.

$$(1+p^n+p^{2n}+\ldots+p^{\infty})\times(1+p^m+p^{2m}+\ldots+p^{\infty})$$

$$= .\overbrace{10\ldots0}^{n-1}\overbrace{10\ldots0}^{n-1}\ldots \times .\overbrace{10\ldots0}^{m-1}\overbrace{10\ldots0}^{m-1}\ldots$$

a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	…	$a_k\ldots$
$\times b_1$	b_2	b_3	b_4	b_5	b_6	b_7	b_8	…	$b_k\ldots$
a_1b_1	a_2b_1	a_3b_1	a_4b_1	a_5b_1	a_6b_1	a_7b_1	a_8b_1	…	$a_kb_1\ldots$
	a_1b_2	a_2b_2	a_3b_2	a_4b_2	a_5b_2	a_6b_2	a_7b_2	…	$a_{k-1}b_2\ldots$
		a_1b_3	a_2b_3	a_3b_3	a_4b_3	a_5b_3	a_6b_3	…	$a_{k-3}b_3\ldots$
				.....					
									$a_1b_k\ldots$

Ignore the carry, the k*th* products digit is $a_kb_1 + a_{k-1}b_2 + \ldots + a_1b_k$

Set $(k'+1)\times LCM(n,m) > k > k'\times LCM(n,m)$ let $LCM(n,m) = \ell$,

$$\underbrace{a_kb_1 + a_{k-1}b_2 + \ldots + a_{k-\ell+1}b_\ell}_{1} + \underbrace{a_{k-\ell}b_{\ell+1} + a_{k-\ell-1}b_{\ell+2} + \ldots + a_{k-2\ell+1}b_{2\ell}}_{2} + \ldots + \underbrace{a_{k-k'\ell}b_{k'\ell+1} + \ldots + a_1b_k}_{3}$$

$\because a_{k+i} = a_{k-\ell+i} = \ldots = a_{k-k'\ell+i}$ & $b_i = b_{\ell+i} = \ldots = b_{k'\ell+i}$

$\therefore$ part 1 can be rewrite as $a_{k-k'\ell}b_1 + a_{k-k'\ell-1}b_2 + \ldots + a_1b_{k-k'\ell} + a_\ell b_{k-k'\ell+1} + \ldots + a_{k-k'\ell+1}b_\ell$

part 2 can also be rewrite as $a_{k-k'\ell}b_1 + a_{k-k'\ell-1}b_2 + \ldots + a_1b_{k-k'\ell} + a_\ell b_{k-k'\ell+1} + \ldots + a_{k-k'\ell+1}b_\ell$

part 3 can be rewrite as $a_{k-k'\ell}b_1 + a_{k-k'\ell-1}b_2 + \ldots + a_1b_{k-k'\ell}$

We can find that part 1 equal to part 2, and because there are *LCM(n,m)* items in this part, we must get $a_1 \times b_1$ once, which is the only item has the value 1, and all other items' value is 0, that is the result of part 1 is 1, so far as part 2.

Part 3 is the actually the front part of part 1, that means the value of part 3 may be 1 or 0.

Base on this discovery, we can divide the configuration into $LCM(n,m)\times LCM(n,m)$ modules. Let's take $100100100\ldots\times101010\ldots(p=5)$ as an example.

LCM(2,3)=6, so we divide the module in 6×6

.100100100100100100100100100100...

×.101010101010101010101010101010...

1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1
			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
				1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0
					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
						1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0
							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
								1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1
									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
										1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0
											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
												1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0
													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
														1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1
															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
																1	0	0	1	0	0	1	0	0	1	0	0	1	0
																	0	0	0	0	0	0	0	0	0	0	0	0	0
																		1	0	0	1	0	0	1	0	0	1	0	0
																			0	0	0	0	0	0	0	0	0	0	0
																				1	0	0	1	0	0	1	0	0	1
																					0	0	0	0	0	0	0	0	0
																						1	0	0	1	0	0	1	0
																							0	0	0	0	0	0	0
1	0	1	1	1	1	2	1	2	2	2	2	3	2	3	3	3	3	4	3	4	4	4	4	5	4	5	5	5	5

We can find two type of modules here, one is:

1	0	0	1	0	0
	0	0	0	0	0
		1	0	0	1
			0	0	0
				1	0
					0

This one is corresponding to the part3;

The other one is:

1	0	0	1	0	0
0	0	0	0	0	0
0	0	1	0	0	1
0	0	0	0	0	0
0	1	0	0	1	0
0	0	0	0	0	0

This one is corresponding to the part1 part2 and so on, we can accumulate every line of this module, the result is always 1, which is the previous conclusion we draw for part 1.

Ignore the carry, assume that c_k is the *kth* digit of the product, we can find that $c_{k+t} = c_k + 1$

Now let's continue to get the result:

101111 212222 323333 434444 545555 656666 767777 878888 989999…

101111 212222 323333 434444 001111 212222 323333 434444 001111…

$$100100100\ldots \times 101010\ldots$$
$$= \overline{10111121222233333444440}$$

Conclusion: In multiplication, the length of periodic part of the product is:

$$LCM(m,n)\times(p-1), \qquad (2)$$

where m and n are the length of periodic part of the two multipliers, p is the prime.

The length of periodicity of the resulting p-adic sequence can be very large from (2). But if we should represent all the p-adic sequences with a complete period during all the calculations, we will definitely carry out all the arithmetic operations exactly.

4. Computational Algorithm and Software for Generating *P*-adic Cyclic Codes

A.R.Calderbank and N.J.A. Sloane [2] "*Modular and p-adic cyclic codes*", studied how to lift from binary cyclic Hamming code of length 7 to octacode, which is equivalent to the binary nonlinear Mordstrom-Robinson code, and provided a general structure of cyclic codes over Z_8, Z_{16}, ..., then to the 2-adic integers $Z_2\infty$. That is to say, all the cyclic codes share a common generator polynomial $X^3 + x^2 + (\lambda-1)x - 1$, which satisfies $\lambda^2 - \lambda + 2 = 0$, λ is a 2-adic number.

The purpose of our study is to provide an alternative solution to generate the p-adic cyclic codes, which can be implemented efficiently using computer science methodology, and we developed software to generate p-adic cyclic codes.

Theoretical Background

According to the definition of cyclic code, a subset S of Z_q^n is cyclic if $(a_{n-1},a_0,a_1,\ldots,a_{n-2}) \in S$ whenever $(a_0,a_1,\ldots,a_{n-2},a_{n-1}) \in S$. A linear code C is called a cyclic code if C is a cyclic set.

We also know that, $R = Z_q[X]/(X^n-1)$ forms a polynomial ring. By defining the following map between Z_q^n and $Z_q[X]/(X^n-1)$,

$\pi: Z_q^n \rightarrow Z_q[X]/(X^n-1)$, $(a_0,a_1,\ldots,a_{n-1}) \mapsto a_0+a_1x+\ldots+a_{n-1}x^{n-1}$ (3)

We can set up a one to one correspondence between an element of space Z_q^n and an element polynomial of ring $Z_q[X]/(X^n-1)$.

Connect ideals of ring $Z_q[X]/(X^n-1)$ and cyclic codes contained in Z_q^n by the following theorem:

Theorem 1 Let π be the linear map defined in (3). Then a nonempty subset C of Z_q^n is a cyclic code if and only if $\pi(C)$ is an ideal of $Z_q[X]/(X^n-1)$. (Proof in [4], p.136).

For the convenience of our discussion, we give the following definition of the generator polynomial of a cyclic code.

Definition 1 The unique monic polynomial of the least degree of a nonzero ideal I of $Z_q[X]/(X^n-1)$ is called the generator polynomial of I. For a cyclic code C, the generator polynomial of $\pi(C)$ is also called the generator polynomial of C.

By using this definition, we can set up an one-to-one correspondence between the cyclic codes in Z_q^n and the monic divisors of $X^n-1 \in Z_q[x]$.

To be precise, each monic divisor of X^n-1 is a generator polynomial of a cyclic code in Z_q^n.

For example, by factorizing the polynomial $x^6 - 1 \in Z_2[x]$:

$$X^6 - 1 = (1 + x)^2(1 + x + x^2)^2$$

We can list all the monic divisors of $x^6 - 1$:

1, $1 + x$, $1 + x + x^2$,
$(1 + x)^2$, $(1 + x)(1 + x + x^2)$, $(1 + x)^2(1 + x + x^2)$,
$(1 + x + x^2)^2$, $(1 + x)(1 + x + x^2)^2$, $1 + x^6$.

These nine monic divisors are related to nine ideals of ring $Z_2[x]/X^6-1$, and thus related to nine cyclic codes of length 6. Based on the map π, we can generate all these cyclic codes.

For instance, we can get the cyclic code corresponding to the polynomial $(1 + x + x^2)^2$ as its generator by the following way:

Step 1. Expand the generator polynomial, $(1 + x + x^2)^2 = 1 + x^2 + x^4$.

Step 2. Multiply the generator polynomial with all the polynomials within the ring $Z_2[x]/X^6-1$, and get the following ideal I:

$$\{0, 1 + x^2 + x^4, x + x^3 + x^5, 1 + x + x^2 + x^3 + x^4 + x^5\}.$$

Step 3. Using map π, we get the corresponding cyclic code:

$$\{000000, 101010, 010101, 111111\}.$$

In general, we denote *[n,k,d]* to describe a linear code and naturally a cyclic code, where *n* is the length of the codeword, *k* is the number of the base vectors, and *d* is the least distance between any two code words.

The following theorem can relate *[n,k,d]* of a cyclic code to the features of the generator polynomial of an ideal I of ring $Z_q[X]/(X^n-1)$.

Theorem 2 Let *g(x)* be the generator polynomial of an ideal of $Z_q[X]/(X^n-1)$. Then the corresponding cyclic code has dimension *k* if the degree of *g(x) is n-k*, and the length of the codeword is *n*.

In the above example, the generator polynomial is $g(x) = 1 + x^2 + x^4$, and $n = 6$, $k = 6-4 = 2$.
This relates to a *[6,2,3]* cyclic code. The length of the codeword is 6, and this cyclic code will have $2^k = 2^2 = 4$ code words.

The following theorem decides the number of cyclic codes:
Theorem 3 Let $x^n - 1 \in Z_q[x]$ have the factorization

$$x^n - 1 = \prod_{i=1}^{r} p_i^{ei}(x),$$

where $p_1(x), p_2(x), \ldots, p_r(x)$ are distinct monic irreducible polynomials and $e_i \geq 1$ for all $i = 1,2,\ldots,r$. Then there are $\Pi_{i=1}^{r} (e_i+1)$ cyclic codes of length n over Z_q.
The key point of finding all the cyclic codes over Z_q^n is the factorization of X^n-1 over Z_q. We will discuss our method to factorize the polynomial X^n-1 over Z_q.

Mechanism to Generate *p*-adic Cyclic Codes
The idea of generating *p*-adic cyclic codes is given by the following theorem:
Theorem 4 Let $q = p^a$, $1 \leq a \leq \infty$. *If* $g_1(x) \in Z_p[x]$ is a monic irreducible divisor of x^n-1 over Z_p, then there is a unique monic irreducible polynomial $g_a(x) \in Z_q[x]$ which divides x^n-1 over Z_q and is congruent to $g_1(x)$ mod *p*.

The complete proof can be found in [2]. If we define a cyclic code of length *n* over Z_p , we can get another cyclic code of Z_{p2} whose generator polynomial is obtained by lifting the generator polynomial of the first cyclic code to Z_{p2} . And then we can continue this way to get the cyclic code over Z_{pa}, where $1 \leq a \leq \infty$.

An Alternative Algorithm for the Generation of *p*-adic Cyclic Codes
Our algorithm has two steps:

a) Factorization of X^n-1 over Z_2;

b) Lifting the generator polynomial over Z_2r and then to Z_2r+1.

Step a) is the foundation of our implementation. In order to simplify our situation, we will let *n* be a prime, which has the form *8m-1*, so that X^n-1 can always be factorized over Z_2 in the following form:

$$x^n-1 = (x-1)\pi_1(x)\pi_2(x)$$

where all the factors are irreducible.

Factorization of X^n-1 over Z_2

Since

$$x^n\text{-}1 = (x-1)\pi_1(x)\pi_2(x)$$

$$\pi_1(x) = a_0 + a_1x + a_2x^2 + \ldots + a_{n\text{-}1/2}x^{n\text{-}1/2}$$

$$\pi_2(x) = b_0 + b_1x + b_2x^2 + \ldots + b_{n\text{-}1/2}x^{n\text{-}1/2}$$

$a_i, b_i \in Z_2$ ($0 \le i \le n\text{-}1/2$),

So

$$x^n\text{-}1 = (x-1)(1 + x + x^2 + \ldots + x^{n\text{-}1}) = = (x-1)\pi_1(x)\pi_2(x)$$

Let

$$a_0b_0 = 1 \pmod 2,$$
$$a_0b_1 + a_1b_0 = 1 \pmod 2,$$
$$\ldots$$
$$a_0b_{n\text{-}1/2} + a_1b_{(n\text{-}1/2)\text{-}1} + \ldots + a_{n\text{-}1/2}b_0 = 1 \pmod 2,$$
$$a_1b_{n\text{-}1/2} + a_2b_{(n\text{-}1/2)\text{-}1} + \ldots + a_{n\text{-}1/2}b_1 = 1 \pmod 2,$$
$$\ldots$$
$$a_{n\text{-}1/2}b_{n\text{-}1/2} = 1 \pmod 2.$$

And the vector (a0,a1,...,an-1/2,b0,b1,...,bn-1/2) which satisfies all the equations above will be the coefficients of irreducible polynomials $\pi_1(x)$ and $\pi_2(x)$.

Therefore, it is easy for computers to enumerate from (0,0,0,...,0) to (1,1,1,...,1); and because x^n-1 has the unique factorization, so the computer can stop wherever it finds a solution.

Lifting the Generator Polynomial over Z_2r to Z_2r+1

From theorem 1, we know that for every monic irreducible divisor $h_1(x)$ of X^n-1 over Z_p, there is a unique monic irreducible polynomial in $Z_{pa}[x]$ which divides X^n-1 over Z_{pa} and is congruent to $h_1(x)$ mod p. Let

$$x^n\text{-}1 = (x-1)\pi_1(x)\pi_2(x);$$

Assume

$$\pi_1(x) = f_21(x),\ \pi_2(x) = g_21(x);$$

Let

$$f_22(x) = f_21(x) + 2*(c_0 + c_1x + \ldots + c_{n\text{-}1/2}x^{n\text{-}1/2}),$$

$$g_22(x) = g_21(x) + 2*(d_0 + d_1x + \ldots + d_{n\text{-}1/2}x^{n\text{-}1/2}),$$

$$0 \le ci, di \le 1,\ 0 \le i \le n\text{-}1/2.$$

We have

$$f_22(x) \equiv f_21(x) \pmod 2,\ g_22(x) \equiv g_21(x) \pmod 2.$$

Let

$$f_2 2(x) = c'_0 + c'_1 x + \ldots + c'_{n-1/2} x^{n-1/2},$$
$$g_2 2(x) = d'_0 + d'_1 x + \ldots + d'_{n-1/2} x^{n-1/2}.$$

Again, we can use computer to enumerate vector $(c_0,c_1,\ldots,c_{n-1/2},d_0,d_1,\ldots,d_{n-1/2})$ from $(0,0,\ldots,0)$ to $(1,1,\ldots,1)$ to find a value which can satisfy the following equation group:

$$c'_0 d'_0 = 1 \pmod{2^2},$$
$$c'_0 d'_1 + c'_1 d'_0 = 1 \pmod{2^2},$$
$$\ldots$$
$$c'_0 d'_{n-1/2} + c'_1 d'_{(n-1/2)-1} + \ldots + c'_{n-1/2} d'_0 = 1 \pmod{2^2},$$
$$c'_1 d'_{n-1/2} + c'_2 d'_{(n-1/2)-1} + \ldots + c'_{n-1/2} d'_1 = 1 \pmod{2^2},$$
$$\ldots$$
$$c'_{n-1/2} d'_{n-1/2} = 1 \pmod{2^2}.$$

The solution will be the unique factorization of x^n-1 over 2^2, and after getting $f_2 2(x)$ and $g_2 2(x)$, we can continue this step to get $f_2 a(x)$ and $g_2 a(x)$, ($2 \le a \le \infty$).

Software

The following are some screenshots, which can demonstrate our programs:

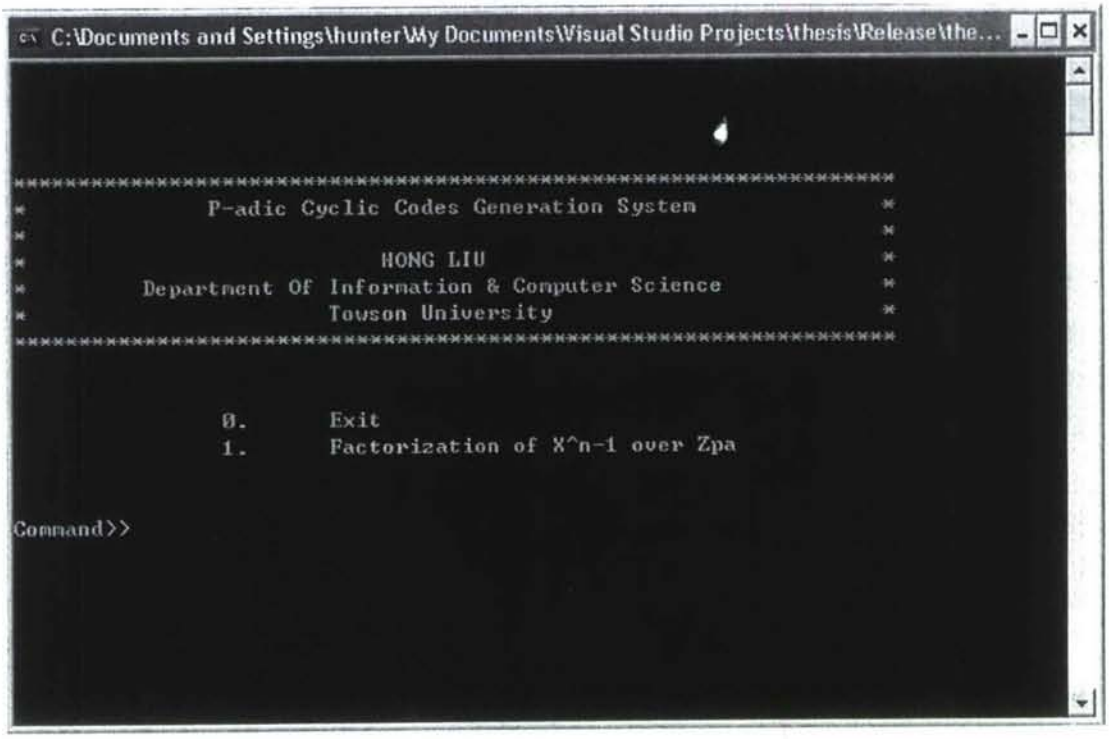

Figure 3. Main Menu

Figure 4 is the screenshot when $p = 2$, $n = 7$, $r = 8$; p should be a prime number, n should be a prime number which has the form $8m-1$, r could be any integer larger than zero. We can find that it has the same result as in *A.R.Calderbank*'s paper.

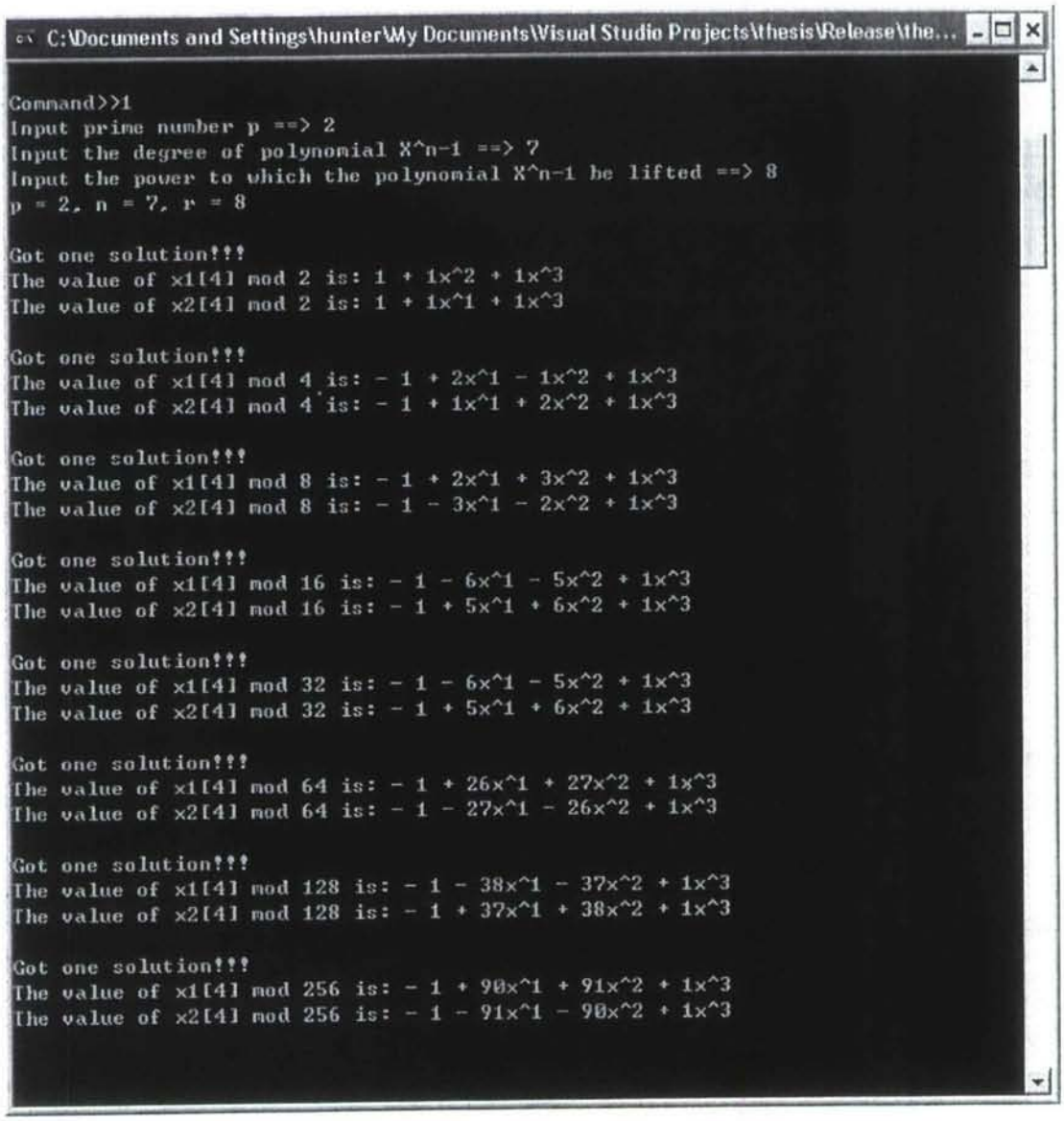

Figure 4. Output When $p = 2$, $n = 7$, $r = 8$

Figure 5 is the screenshot when $p = 3$, $n = 11$, $r = 8$:

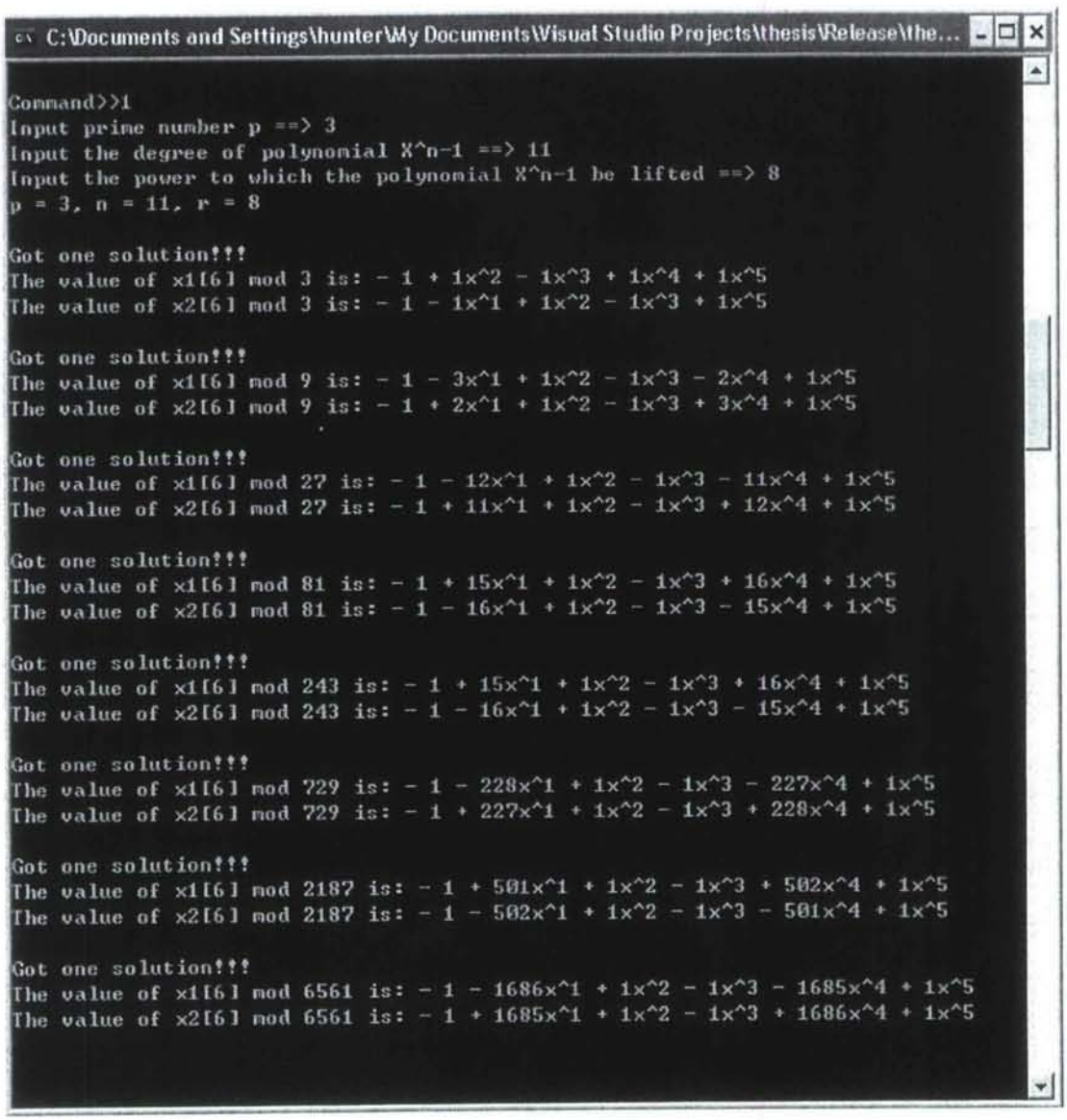

Figure 5. Output When $p = 3$, $n = 11$, $r = 8$

Figure 6 is the screenshot when $p = 2$, $n = 23$, $r = 8$:

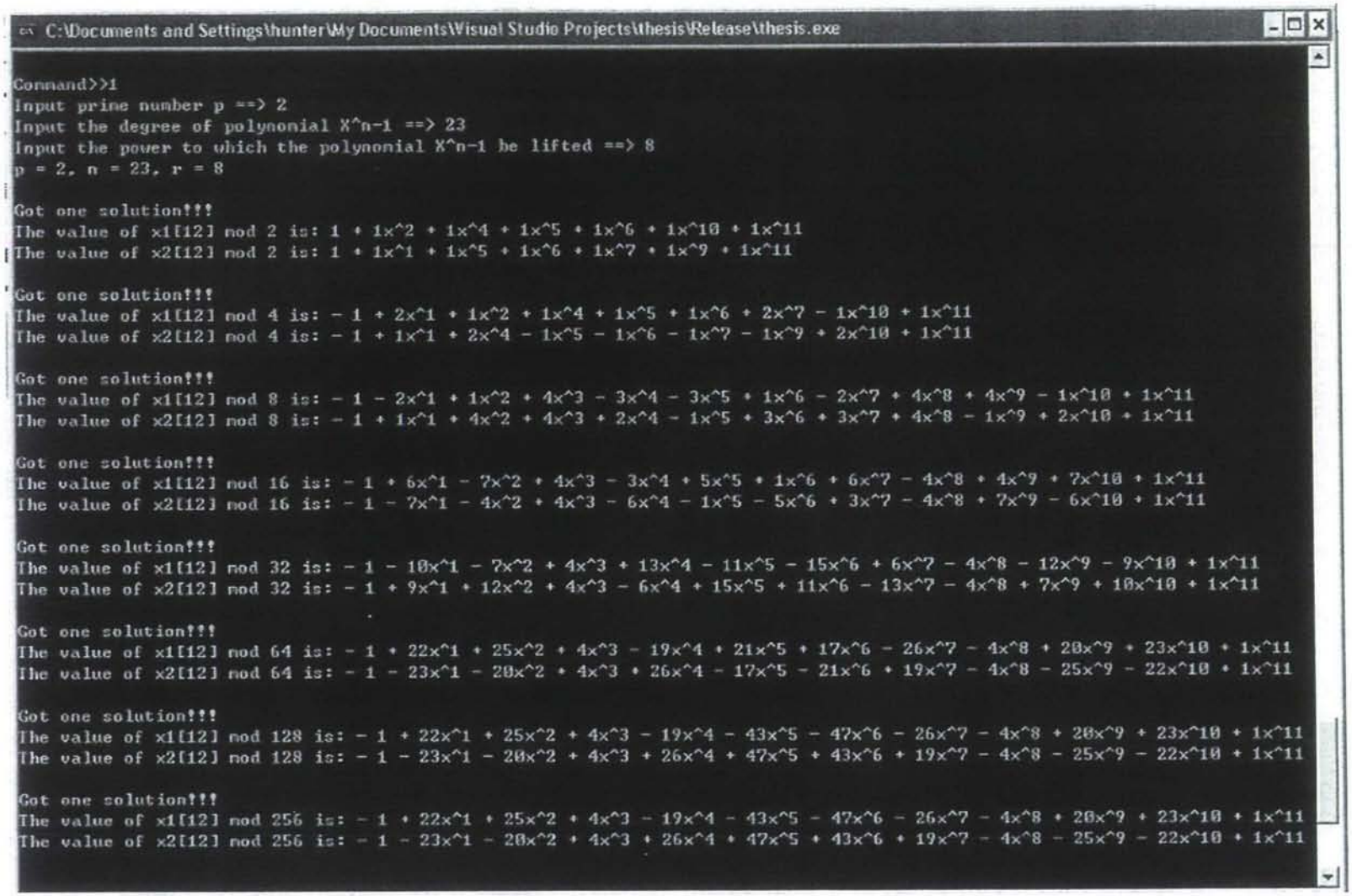

Figure 6. Output When $p = 2$, $n = 23$, $r = 8$

The main function of our software is the factorization of polynomial of X^n-1. From the coding theory, we know that cyclic codes have many good characteristics for error-correction and cryptography. It may have some merit to investigate the p-adic cyclic codes. In this study, by setting up a one-to-one correspondence between cyclic codes and ideals of a polynomials ring, we can use algebraic methods to explore the features of p-adic cyclic codes.

In the future, we may use this software as a tool to study the features of p-adic cyclic codes, and/or use the software to auto-generate cyclic codes in any application.

References

[1] Dixon, J. "Exact Solution of Linear Equations Using P-adic Expansions", Numerische Mathematik 40, 137-141 (1982) Springer-Verlag.

[2] A.R.Calderbank and N.J.A.Sloane, "Modular and p-adic cyclic codes", arXiv:CO/0311319v1, 18 Nov 2003.

[3] Shoup, V *NTL library* at: http://shoup.net/ntl/

[4] San Ling and Chaoping Xing, "Coding Theory: a first course", Cambridge University Press, 2004

[5] Kornerup, P. and Gregory, R.T. "Mapping Integers and Hensel Codes onto Farey Fractions", BIT 23 (1983), 9-20.

[6] Krishnamurthy, F. V. "Matrix Processors Using P-adic Arithmetic for Exact Linear Computations", IEEE Transactions on Computers, vol. C-26, No. 7, July 1977.

[7] Vladimirov, V.S., Volovich, I.V. and Zelenov, E.I. *P-adic Analysis and Mathematical Physics,* Series on Soviet & East European Mathematics – Vol. 1.World Scientific 1993.

[8] Lu, C. and An, M. "Final Report of Simulation of Quantum Time-Frequency Transform Algorithms", (**FA9550-04-1-0406**), 2004.

[9] Lu, C. and An, M. "Final Report of A Computational Library Using *P*-adic Arithmetic for Exact Computation with Rational Numbers in Quantum Computing", (**FA9550-05-1-0363**), 2005.